**09 - Unidad Final**

*-Code Katas and Test-Driven Development - RectangleKata and FizzBuzzKata*

*-Real-world coding to complete the (anything but) "Simple Puzzle" program*

## **Preparación para enseñar este curso**

|  |  |
| --- | --- |
| :hourglass: **Instala** el [TKPJava](https://github.com/swandarina/TKP.Java) Material Didáctico | **1.** Rectángulo Kata - [Respuesta](https://github.com/swandarina/TKP.WANDARINA.Fuente.Java/blob/master/src/main/java/org/teachingkidsprogramming/recipes/completed/section08events/SimpleBubble.java) |
| :green_book: **Lee** la lección que está en esta página | **2.** Llanta Kata - [Respuesta](https://github.com/swandarina/TKP.WANDARINA.Fuente.Java/blob/master/src/main/java/org/teachingkidsprogramming/recipes/completed/section08events/SimpleBubbleVariation.java) |
| :computer:**Codifica** todas las recetas por ti mismo | **3.** FizzBuzz - [Respuesta](https://github.com/swandarina/TKP.WANDARINA.Fuente.Java/blob/master/src/main/java/org/teachingkidsprogramming/recipes/completed/section08events/SimpleBubbleQuiz.java) |
| :bulb: **Revisa** la [diapositivas](https://drive.google.com/a/wandarina.com/file/d/1MnJj1QvxYOELvR4U_L37ydP8-3-nApIE/view?usp=sharing) del Lenguaje de TKP | **4.** FizzBuzzTDD - [Respuesta](https://github.com/swandarina/TKP.WANDARINA.Fuente.Java/blob/master/src/main/java/org/teachingkidsprogramming/recipes/completed/section08events/ConnectTheDots.java) |
| :fax: **Imprime** los [atajos de teclado](https://docs.google.com/document/d/1tzRac5nk5nkL0RcDQ0RiXQJhbRM5NxkNbmA81Dhb0dM/edit#bookmark=id.rgcmyj1v0fcr) | **5.** FizzBuzz Maestro de Oro - [Respuesta](https://github.com/swandarina/TKP.WANDARINA.Fuente.Java/blob/master/src/main/java/org/teachingkidsprogramming/recipes/completed/section08events/DeepDive08Events.java) |
| :fax: **Imprime** [las hojas de trabajo](https://drive.google.com/drive/folders/1P1eq_7YZl-2DKDitlFsVLftzUzQQfOpY?usp=sharing) | **6.** Rompecabezas Simple - [Respuesta](https://github.com/swandarina/TKP.WANDARINA.Fuente.Java/blob/master/src/main/java/org/teachingkidsprogramming/recipes/completed/section08events/TortoiseMaze.java) |
| :books: Continúa aprendiendo | **7.** Inmersión Profunda 09 - [Respuesta](https://github.com/swandarina/TKP.WANDARINA.Fuente.Java/blob/master/src/main/java/org/teachingkidsprogramming/recipes/completed/section08events/DeepDive08Events.java) |

## **Parte 1 - Rectangle Kata**

Un Kata es un conjunto de instrucciones en español/inglés de nivel avanzado, que puede compararse con los requerimientos que demandan la creación de algún software. Como por ejemplo, 'compilar una aplicación para que haga x, y, z, en lugar de crear una instancia de la clase Form ', etc.

Para enseñar esta receta, haga lo siguiente:

* Lea las instrucciones (requerimientos) en voz alta:

|  |
| --- |
| // Draw a rectangle on the board (any size, shape and color) // Write EACH step to code the rectangle in English FIRST   (in Eclipse using code comments) // Determine the ORDER to code each line  // Run the program to see the result  // Refactor to remove duplication |

|  |
| --- |
| // Dibuja un rectángulo en el tablero (cualquier tamaño, forma y color)  // Escribe CADA paso para codificar el rectángulo en español/inglés PRIMERO  (en Eclipse, usando comentarios de código)  // Determine el ORDEN para codificar cada línea  // Ejecute el programa para ver el resultado  // Refactorice para eliminar duplicaciones |

La razón de este ejercicio es que este tipo de deconstrucción de problemas se acerca a la programación del mundo real, ya que imita el proceso siguiendo los pasos que normalmente en la práctica se usarían, de la siguiente manera:

* Obtenga un **objetivo de negocio**, defínalo y póngase de acuerdo sobre él = 'dibuje un tipo de rectángulo'
* **Dibuje** o bosqueje el objetivo del negocio (la comprensión más común para los propietarios del producto, desarrolladores y diseñadores) = dibujando realmente el rectángulo
* Escriba **pasos discretos en español/inglés** PRIMERO - los desarrolladores son MÁS FLUIDOS en español/inglés que en cualquier idioma de codificación = exprese intención claramente en el idioma nativo primero
* **Determine el orden** de código para que los desarrolladores puedan verificar que un paso se haya codificado correctamente = deconstruir el problema para que la lógica sea capturada correctamente
* **Codifique y verifique** = haga todos los pasos anteriores ANTES de codificar en busca de calidad
* **Refactorice** para eliminar el código duplicado y mejorar la legibilidad = código de mejor calidad.

## **Parte 2 - Receta: WheelKata**

Otro **Kata**, o conjunto de instrucciones en español/inglés de nivel avanzado, el cual hace que los estudiantes trabajen más como verdaderos desarrolladores de software. Para enseñar esta receta, haga lo siguiente:

* Lea las instrucciones (requisitos) en voz alta:

|  |
| --- |
| // Dibujar una rueda en el tablero (de cualquier tamaño, forma y color)  // Escribe CADA paso para codificar la rueda en español/inglés PRIMERO  (en Eclipse usando los comentarios de código)  // Determine el ORDEN para codificar cada línea  // Ejecuta el programa para ver el resultado  // Refactorice para eliminar duplicaciones |

|  |
| --- |
| // Draw a wheel on the board (any size, shape and color) // Write EACH step to code the wheel in English FIRST   (in Eclipse using code comments) // Determine the ORDER to code each line  // Run the program to see the result  // Refactor to remove duplication |

NOTA: Los estudiantes necesitarán implementar un método **principal** en su código como mínimo para resolver este desafío. Probablemente volverán a re-factorizar para tener múltiples métodos.

## **Parte 3 - Receta: FizzBuzz.java & Receta: FizzBuzzTDD.java**

Otro **Kata**, o conjunto de instrucciones en español/inglés de nivel avanzado, el cual hace que los estudiantes trabajen más como verdaderos desarrolladores de software. Para enseñar esta receta, haga lo siguiente:

* Lea las instrucciones (requisitos) en voz alta:

|  |
| --- |
| // para los números enteros del 1 al 100, imprima ese número, o,  // si ese número es divisible por 3, imprima la palabra 'Fizz',  // si ese número es divisible por 5, imprima la palabra 'Buzz',  // si ese número es divisible por 3 o 5, imprima la palabra 'FizzBuzz'\_ |

|  |
| --- |
| //for the whole numbers from 1 to 100, print either that number, or,  //if that number is evenly divisible by 3, then print the word 'Fizz',  //if that number is evenly divisible by 5, then print the word 'Buzz',  //if that number is evenly divisible by either 3 or 5, then print the word 'FizzBuzz'\_ |

* Guíe a la clase en una discusión sobre cómo traducirán este Kata en líneas en español/inglés (que luego serán traducidas a código, una línea a la vez). Haga preguntas como "¿dónde empezarás a codificar esto y por qué?" Aliente a los estudiantes a ingresar comentarios en ESPAÑOL/INGLÉS ANTES de que comiencen a escribir el código. Ayude a los estudiantes a enfocarse en los requisitos del USUARIO, haciéndoles preguntas tales como *'cuando un USUARIO ingresa un número, entonces ¿ese USUARIO obtiene un tipo particular de resultado?'*

Puede enseñar esto como un Kata normal primero, y luego como un estilo TDD, o puede enseñar esta receta como un kata de estilo TDD para comenzar, dependiendo de la experiencia de tus alumnos.

**El desarrollo basado en pruebas** es un estilo de codificación en el que el desarrollador escribe el caso de prueba ANTES de escribir el código de implementación. TDD usa pruebas unitarias para probar cada implementación de la lógica del programa. Cada prueba de la unidad TDD se codifica como un método con el atributo de método @Test.

Como hemos hecho en las secciones de Inmersión Profunda del material didáctico de TKP, para proceder con cada prueba, los estudiantes deben ejecutarla y deberían ver un error en el resultado (o barra roja). El trabajo de los estudiantes es escribir el código que implementará la lógica deseada en el archivo de clase, en este caso 'FizzBuzz.java' y el método (FizzBuzz.convert) que producirá la salida/resultado esperado por el usuario.

Un aspecto importante de la enseñanza de TDD es guiar al estudiante a 'cortar en rodajas' o escribir un escenario ‘fácil de probar’ a la vez. Después de que escriban la prueba, la ejecuten y vean que la prueba falla, el alumno escribirá el código de implementación para pasar la prueba.

Asegúrese de basar cada caso de prueba (método de prueba) en un único ESCENARIO DE USUARIO, por ejemplo, '\_si envía el número uno, obtiene el número uno'\_, etc.; en lugar de construir andamios, como por ejemplo, ' \_pruebe que podemos crear una clase llamada FizzBuzz\_ '.

El facilitador debe guiar a las parejas para que creen un código auxiliar de prueba JUnit, con un único método de @Test, para probar el primer caso (IfOneReturnOne):

|  |
| --- |
| @Test  public static void TestOneReturnsOne()  {  String result = FizzBuzz.convert(1);  Assert.areEqual("1", result);  } |

Continúe con el próximo caso de prueba, (IfTwoReturnTwo).

A continuación, aumente la complejidad para escribir un caso de prueba para (IfThreeReturnFizz). Los estudiantes podrían tener dificultades al intentar convertir un número entero en una cadena de texto en el objeto FizzBuzz. Hay múltiples formas de realizar la conversión de cadenas de texto. Recomendamos el caso más escueto:

|  |
| --- |
| return "" + yourNumber; |

Alternativamente, puede usar la conversión más detallada mostrada a continuación:

|  |
| --- |
| return String.valueOf(yourNumber); |

Asegúrese de hacer que los alumnos creen casos de prueba que primero fallen (pídales a los alumnos que realicen la respectiva prueba para verificar). Y luego pida a los estudiantes completen cada paso del ejercicio. Nuevamente, es necesario que los estudiantes ejecuten cuantas veces sea necesario, cada prueba hasta que la logren pasar (o se presente una barra verde). Cuando los estudiantes estén atascados, pídales que hagan doble clic en el detalle de la excepción (panel inferior izquierdo del editor) para abrir los resultados en la herramienta de diferencia de texto predeterminada, de esta forma, ellos podrán ver los detalles de los errores de salida, y podrán corregir (rápidamente) con precisión su código para producir el resultado deseado.

**NOTAS DE LÍNEA**

#### Este kata tiene más de una solución. Pídales a los alumnos que expliquen por qué eligieron codificar la lógica condicional de la forma en que lo hicieron.

### **Recapitulación de Kata de FizzBuzzTDD**

Video para la comprensión del docente.

[](https://www.youtube.com/watch?feature=player_embedded&v=p5EAGD7fOHQ)

## **Parte 4 – Variación: Refactoriza la lógica condicional**

Complete todos los requisitos para FizzBuzz, ahora que ha escrito todas las Pruebas de las Unidades (y también, lo que es más importante, ha escrito el código requerido para que las Pruebas de las Unidades pasen). Para hacer esto, necesitarás hacer lo siguiente:

|  |
| --- |
| // Copia el archivo FizzBuzz.java, renómbralo a FizzBuzz2.java  // Crea un método principal (punto de entrada)  // Dentro de ‘main’ crea un contenedor para tener los resultados / StringBuilder  // Dentro del método principal interno crea una lista de números del 1-100  // Añade cada número convertido a StringBuilder  // Imprime los resultados en la consola, crea una nueva línea para cada número |

|  |
| --- |
| // Copy the FizzBuzz.java file, rename it to FizzBuzz2.java // Create a main method (entry point) // Inside main create a container to hold the results / StringBuilder // Inside main method create list of numbers from 1-100 // Append each converted number to the StringBuilder // Print the results to the console, create a new line for each number |

Una posible solución se muestra a continuación:

|  |
| --- |
| package org.teachingkidsprogramming.sectionLYNNtdd;  public class FizzBuzz2 {  public static void main(String[] args)  {  StringBuilder sb = new StringBuilder();  for (int i = 1; i <= 100; i++)  {  sb.append(convert(i));  }  System.out.print(sb);  }   public static String convert(int i)  {  if (0 == i % 15) { return "\n FizzBuzz"; }  if (0 == i % 5) { return "\n Buzz"; }  if (0 == i % 3) { return "\n Fizz"; }  return "\n " + i;  } } |

NOTA: Hemos incluido una versión muy avanzada de FizzBuzz que utiliza el concepto de prueba de unidad de prueba de 'aprobación' (o Golden Master). Para trabajar con este ejemplo, deberá instalar una herramienta de comparación de archivos, como por ejemplo, “Beyond Compare”.

## **Parte 5 – Receta: Rompecabezas Simple**

Conceptos revisados en esta solución:

1) Uso de objetos, creación de objetos nuevos

2) While Loops

3) Probar ... Adquirir construcción

4) Manipulación de *Arrays*

Esta es la última lección en Java de TKP. Irónicamente se llama 'Rompecabezas Simple', debido a que está lejos de ser simple. En esta lección simulamos el entorno de trabajo de un programador profesional. En otras palabras, la mayoría de los desarrolladores trabajan en bases de códigos existentes, en lugar de codificar proyectos completamente desde cero; también por ejemplo, implementan un API existente, en lugar de codificar nuevas funcionalidades desde cero. En este caso, escribimos un API para el rompecabezas (un solucionador) para que los estudiantes lo implementen.

## **Parte 6 – Examen: Ninguno**

No hay examen para este curso. Es posible que desee ver un video interesante sobre muchos enfoques diferentes para la codificación de FizzBuzz del orador [Kevlin Henney "FizzBuzz Trek"](https://vimeo.com/144335290). Para su información, FizzBuzz es una entrevista común para programadores profesionales. Para ver más ejemplos de código katas, mira este [sitio](http://www.codekatas.org/).

## **Part 7 - Inmersión Profunda: DeepDive08**

Sobre Excepciones

## **Part 8 – Hoja de Trabajo: FizzBuzz**

Los conceptos revisados en esta hoja de trabajo son los siguientes:

1) Tipo de Conversión - Número (int) a Cadena de Texto

2) Prueba de Unidad: estructura de una declaración básica de *Assert*

3) Prueba de Unidad: configuración de prueba y ejecución de prueba, es decir, Assert.areEquals (esperado, real)

4) sintaxis @Test - atribución de método para pruebas de Unidad

5) Puntos de salida del programa

Próximos pasos y recursos adicionales

Después de que los estudiantes completen todas las lecciones de este curso, pueden decir que han culminado el programa en Java de TKP. Como material adicional, podrán encontrar un gran conjunto de ejercicios en el sitio web [exercism.io](http://exercism.io/). Este sitio proporciona una plataforma para continuar aprendiendo a programar, mediante la escritura de código, y el envío de soluciones de código para "recibir la revisión de personas alrededor del mundo". Asegúrese también de leer [la guía de inicio](http://exercism.io/how-it-works/newbie).